Skincare Ontology

Final Project

Sara Abesová[1], Karolína Hajková[2], Yozlem Ramadan[3], and Małgorzata Zdych[4]

[1] saa275, 2735679
[2] kha520, 2728951
[3] yrn420, 2732940
[4] mzh800, 2730740
Vrije Universiteit Amsterdam
Knowledge and Data
Group 31

Table of Contents

List of Figures

1 Introduction

The human face is one of the most crucial parts of one's image, and thus, feeling good about it is vital for the overall well-being of a person. Appropriate treatment maintains the skin in a good condition and can prevent undesirable skin problems now and in the long term. Choosing the correct pipeline of skincare routine can be challenging due to the amount of information available and determining which data is valuable [4]. Personalized guidelines towards healthy skin are essential, as the needs of distinct individuals vary based on their skin concerns. Therefore, the skincare routine is heavily dependent on particular requirements that result from the user's skin type and skin tone. Additionally, one's willingness to maintain a consistent routine plays a key role in the efficacy of the treatment. Thus, the complexity of the routine should also be one of the factors taken into consideration in any recommendation system that is supposed to facilitate one's skincare journey.

2 Goal

The objective of this analysis is to provide insight into the complex sphere of skincare by providing users with personalized routines that will enhance the health of their facial skin. The designed system will present the most suitable products based on individual input parameters. Users will provide the system with information about their skin type, skin tone, residential country, and desired complexity of the overall routine. Providing correct guidelines and the most fitting products to those seeking advice (in the use of facial cosmetic products for treatment) will not only improve one's interest in skin health, but what is more, it is going to establish promising results that are grounded in previous experiences of people with similar visage. The goal output is to recommend 3 most suitable products per category (Cleanser, Moisturizer, Treatment). The number of categories that is outputted depends on the desired user complexity. In addition, the system should output the local Sephora link and possibly physical location. The ultimate goal is to strengthen one's self-assurance and confidence by enhancing the attitude toward their skin.

Designed ontology with personalized routines aims to make the recommendation system beneficial, as well as inclusive. It prioritizes making suitable skincare choices and moreover, focuses on making knowledge concerning skin health as accessible as possible to anyone interested.

3 Stakeholders

The potential stakeholders of the analysis are individuals of all genders, ages, and ethnicities who are interested in enhancing the health of their skin. The recommendation system is beneficial mainly for less experienced or inexperienced users who will be provided with personal guidelines that will navigate them

through this complex field. It will work as an aid tool in establishing a personalized routine for their specific skin type, skin tone, and desired complexity of the overall process.

However, our project leaves space for further improvements, specifically expanding the recommendation tool. The potential stakeholders could be researchers that are interested in improving the client's awareness of the field of facial skincare. Further improvements to the system developed in this paper will be elaborated in Section 12 Limitations and Further Improvements. [2].

4 Design and Walkthrough

The first step of the analysis consisted of making the decision of which path to take within the domain of skincare, as it is broad and complex. The base for the upcoming recommendation system was a skincare ontology that was created by one of the team members earlier in the course. The former ontology was at first focused more on universal products; not taking into consideration one's skin parameters. However, in the process of data preparation, it was adjusted to focus on how distinct individuals could benefit from the use of products personalized for their needs by implementing the differentiation between skin types and tones.

Various obstacles had to be tackled before the actual design of the personal recommendation system could begin. The most challenging segment of the analysis was to find external databases that would match well with the vision of the team and from which newly created ontology could benefit. After extensive research on the web, a CSV file containing information scraped from Sephora's website was found on GitHub platform [3]. This file provided the data on individual reviews of skincare products from the Sephora website. The aforementioned data set became the main and most significant source for the system design, as it provided the team with a comprehensive and extensive base of information that could be reused, processed, and implemented into the final ontology. Although, the file also needed to be prepared before being part of the analysis and also be merged with the base ontology. The irrelevant information included in the CSV set needed to be deleted, for instance, the Eye_Color and Hair_Color columns. After the data was filtered based on its significance, the team ensured that there is no information missing or that some parts of important data were not accidentally removed, so the derived set is an accurate data representation. Afterward, the merging with the base ontology could start, however beforehand the manual check between the consistency of both data sets was performed. The columns that had the same names were assigned to Class Restrictions, as Equivalent Classes, for instance, rdf:Cleanser (external CSV file) to skc:Cleanser (base ontology). When all the classes were prepared and data was modified for the purpose of the project, it could be eventually put in the OntoRefine to be mapped with the base ontology. The graphs were constructed for the specific triples and tested for errors. Once the data was validated, it was also checked for a sufficient quantity of data for the analysis, so later it could be downloaded as Turtle files. Eventually, all nine turtle files derived from the CSV file were imported to the

final default graph, in which base ontology was already mapped. Later in the process, an additional data set was developed - a CSV file. It consisted of the list of countries where Sephora is available connected to the website links. This information was extracted from the main Sephora web page [1]. The CSV file was put in the OntoRefine where RDF mapping was edited, so the triple Country hasSephoraWebPage SephoraPage could be derived. The CSV file did not need extra validation, as it was a simple dataset with two columns. Eventually, the new data set of Sephora stores was also imported to the default graph. In this, the default graph, also including the base ontology and the first external CSV file was finally exported and later reused for analysis and visualization. Moreover, the data about the countries was derived from the DBpedia database - the second external source. The implementation of DBpedia was needed to obtain information about the coordinates of countries, where Sephora is available. More precisely, the capital cities, as only this type of information is provided by DBpedia. The latitude and longitude of capital cities were queried from DBpedia and used for generating the map in visualizations from the folium libraries, for which the coordinates of capital cities were used as a reference to where Sephora shops are located. This type of query were created to support the analysis and provide the users with a clear and accessible overview of the physical Sephora stores' locations. In the later steps, other queries were developed and implemented to create visualizations with panda libraries. Two pie graphs were developed to represent the overview of reviews that distinguished firstly between skin tones and secondly skin types. Additionally, the latter colors of the pie graph were chosen from a color palette, in such a way that they would correspond with the color of one's skin tone. Therefore, the visualizations presented in such a way could clearly show them, how much data there exists for their particular skin color and skin type, so they can themselves see how reliable the results of the recommendation systems are. The visualizations can be seen below.

4.1 Visualisation

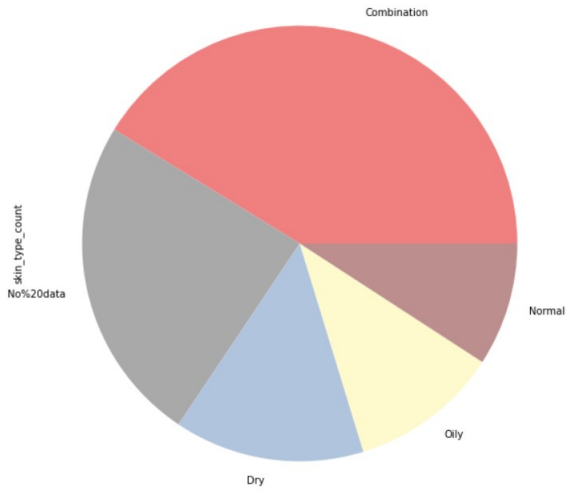


Fig. 1: Pie-graph visualising count based on skin type

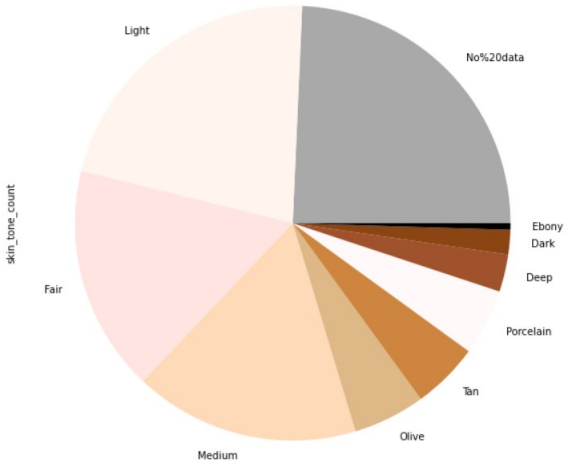


Fig. 2: Pie-graph visualising count based on skin tone

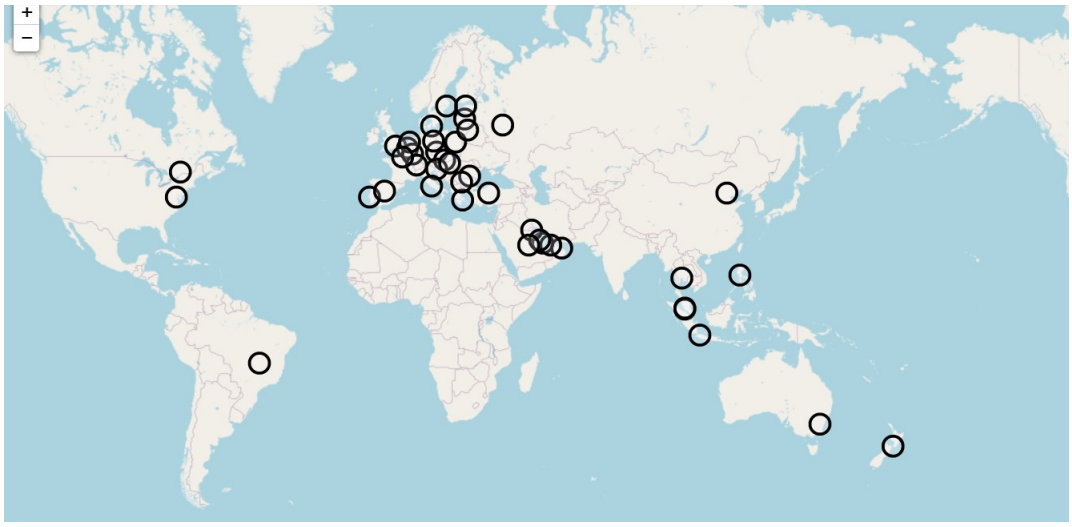


Fig. 3: Map representing countries with Sephora stores

5 Domain and Scope of the Ontology

The domain of the ontology focuses on the facial skincare. The domain's scope covers factors which will allow to make analysis in order to achieve the aim of the project. For instance the information needed from the user to determine recommendation about them includes their skin type, skin tone, number of steps they would prefer for their routine and country where they live. In order to have analysis over these inputs, the designed ontology will have information about suitable products for every skin type, which products can be used by people with specific skin tone, number of products according to the number of steps in the routine and links where to find the products, based of the country they live in. In addition, this information is needed to create visualisations that will illustrate the availability of Sephora stores and show users what type of data the team is extracting to output the suggestion results.

6 Methodology for the Ontology Construction

The ontology has been created according to middle-out approach. First, starting with the base ontology, constructed by one of the teammates, some of the main classes and properties have been already defined. Later, a CVS file with relevant information about the domain was discovered, containing a large data set regarding skincare retrieved from Sephora's website. This new file made the scope of the domain broader, containing more details to expand the initial ontology with more classes and properties stated presented in hierarchical order to illustrate the relations between them. Here the data from external databases

set further direction for the ontology. Further modelling of appropriate classes and properties became necessary due to the new data. Afterwards, additional properties and classes were added to combine the two data sources and unite the ontology. Similar process was employed with the second CVS file, about the countries where Sephora's shops are located. It is clear that the process was iterative, starting with fundamental concepts and working towards both more abstract and specific terms.

7 Conceptualization of the Ontology

Designed ontology consists of a total of 36 classes, some being sub-classes of others. The ontology specifies 6 object properties and 6 data properties.

7.1 Classes

- skc:Country
- skc:Brand
- skc:Category
  - rdf:Cleanser
  - rdf:Moisturizer
  - rdf:Treatment
    - skc:Exoliant
      - skc:ChemicalExoliant
      - skc:PhysicalExoliant
    - skc:Serum
    - skc:Toner
  - skc:Cleanser
  - skc:Moisturizer
    - skc:SPF
- skc:Product
  - skc:Combination_Skin_Products
  - skc:Dry_Skin_Products
  - skc:Normal_Skin_Products
  - skc:Oily_Skin_Products
- skc:Rating_Stars
- skc:Review_Id
- skc:Skin_Tone
  - skc:Dark
  - skc:Deep
  - skc:Ebony
  - skc:Fair
  - skc:Light

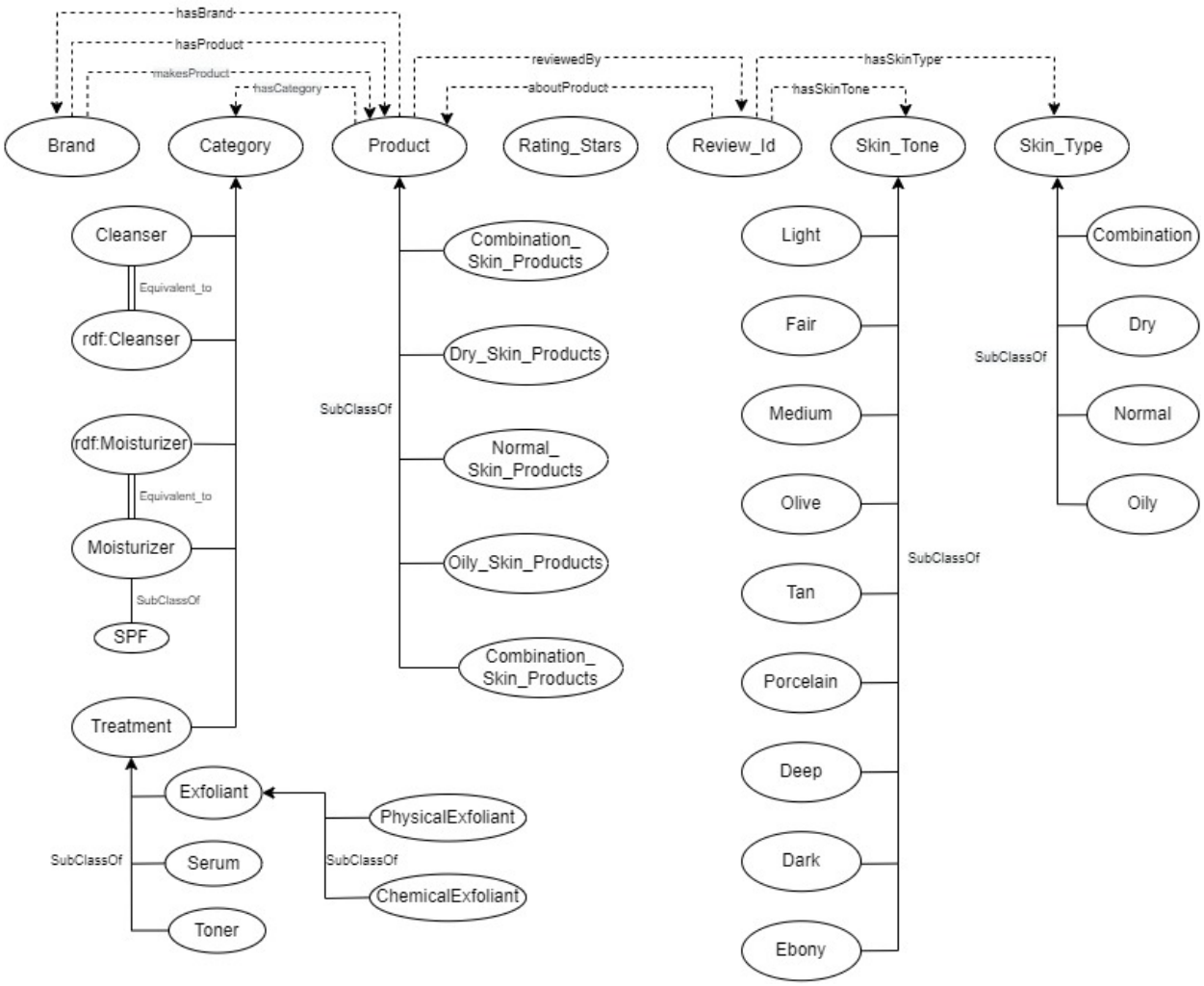


Fig. 4: Ontology Conceptualisation represented in a graph

- skc:Medium
- skc:Olive
- skc:Porcelain
- skc:Tan

– skc:Skin_Type
  - skc:Combination
  - skc:Dry
  - skc:Normal
  - skc:Oily

7.2 Object Properties

The table below displays how a particular instance relates to a class with the use of various object properties.

Object Properties		
Subject	Object Property	Instance
Product	hasBrand	Brand
Review_Id	hasSkinType	Skin_Type
Review_Id	hasSkinTone	Skin_Tone
Review_Id	aboutProduct	Product
Product	hasCategory	Category
Product	hasReviewId	Review_Id

In the ontology Product hasCategory Category was not implemented, as it is presented in the table above. The team found out about the mistake too late to modify it in the OntoRefine. Thus, ideally instead of Product rdf:type Category in our ontology, the solution Product hasCategory Category would be applied. It does not make ontology inconsistent, only slightly illogical, as every product is rdf:type Category due to rdfs rules set.

7.3 Data Type Properties

The table below presents the relation between a class and a literal value, such as integer.

Data Properties		
Subject	Data Property	Data Type
Product	hasRating	"Rating_Stars" (literal)
Product	hasCombinationScore	"CombinationScore" (decimal)
Product	hasNormalScore	"NormalScore" (decimal)
Product	hasDryScore	"DryScore" (decimal)
Product	hasOilyScore	"OilyScore" (decimal)
Country	hasSephoraWebPage	"SephoraPage" (literal)

7.4 Reuse of Existing Ontology's

At the very beginning of the project, the already existing ontology was used and also prepared for further modifications, according to the other external sources. A CSV file from the GitHub [3] was found and reused, as it was written about a similar topic of skincare recommendations. It was later on the base of the final ontology. The particular classes, data and object properties were based on the columns from the CSV file, some of them were also reused from the base ontology and adjusted accordingly in both data sets. The classes such as Hair_Color and Eye_Color were removed, as they did not contribute to the skincare recommendation system, and thus they did not serve the purpose of the project. Some of the object properties, for instance "skc:isUsedBefore", "skc:isUsedAfter", "skc:cannotBeUsedWith", "skc:isCompatibleWith" and some of data properties: "skc:hasPrice" and "skc:hasTitle" from the base ontology were removed. Moreover, the class Category was taken from the CSV file, as it was one of the columns, in which the author made a distinction between the types of products, either Moisturizer, Cleanser, or Treatment. Similar differentiation was implemented in the base ontology, however, the class Treatment was not used there initially, but the classes Exfoliant, with its two subclasses Physical Exfoliant and Chemical Exfoliant, as well as class Serum and Toner. Therefore, the team chose to apply it in the final merged ontology and add Exfoliant, Serum and Toner as subclasses of Treatment. Such a modification can be visible in the final ontology.

7.5 Reuse of Existing Vocabulary

Two of the vocabularies that we reused are geo and dbo. The geo vocabulary focuses on queries about spatial information, for instance, places, kilometers, or coordinates. This extension for SPARQL was used to access the geo:lat and geo:long of capital cities that were needed to generate the map for the visualizations. However, to access the coordinates the dbo vocabulary was also needed. SPARQL query against the DBpedia, which is an open data catalog, from which we acquired the subjects – countries for our ontology, for instance, dbr:Czech_Republic. Consecutively, we used them to access the capital cities that had a predicate geo:long and geo:lat. In this way, the team could retrieve the data about coordinates by writing the query with the use of dbo and geo. It was later implemented to derive the map visualization.

7.6 Class Restrictions

- skc:Product
  - Subclass: Oily_Skin_Products
    - Equivalent To:
      - hasOilyScore value 5
  - Subclass: Dry_Skin_Products
    - Equivalent To:
      - hasDryScore value 5
  - Subclass: Combination_Skin_Products
    - Equivalent To:
      - hasCombinationScore value 5
  - Subclass: Normal_Skin_Products
    - Equivalent To:
      - hasNormalScore value 5
- Category:
  - Subclass: skc:Cleanser
    - Equivalent To:
      - rdf:Cleanser
  - Subclass: skc:Moisturizer
    - Equivalent To:
      - rdf:Moisturizer

Additionally, in the final ontology, skc:Toner, skc:Serum and skc:Exfoliant have been restricted as sub-classes of the class skc:Treatment, instead of the initial notion of using "Equivalent To" restriction. All of their instances can be considered as treatments, thus, such adjustments were relevant. By implementing these changes, the team was able to successfully merge the ontologies, ensuring that the products from the base ontology are assigned to the correct type of treatment.

8 Integration of External Datasets

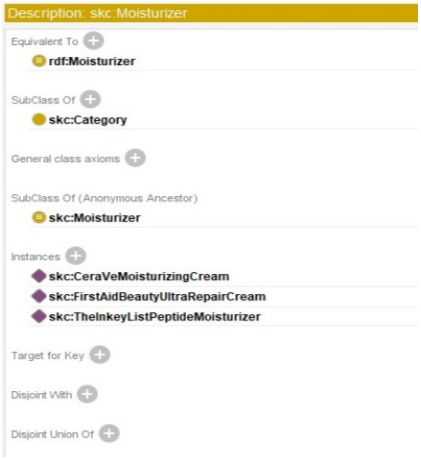
Designed system combined information from multiple external and internal datasets. All of these were chosen attentively, arranging profitable information for the final ontology.

- CSV file on GitHub
  - From this CSV file we extracted the following columns:
    - Skintone, Skintype, Column, Rating_Stars, Product, Brand, Category
- DBpedia page with an instance of a country
  - External SPARQL endpoint
    - This dataset was added to the ontology via a SPARQL query in GraphDB, extra editing was not needed, as it was already in Turtle syntax. The countries, for example dbr:Czech_Republic was mapped to the ontology. It was later reused to obtain latitude and longitude of its capital city, in this case dbr:_Prague that was implemented in the visualisations.
- Sephora website of all countries
  - This information was derived from the Sephora website, where the data about the stores' locations is available. Later it was converted to the CSV file, with two columns: Country and the corresponding Sephora Link.

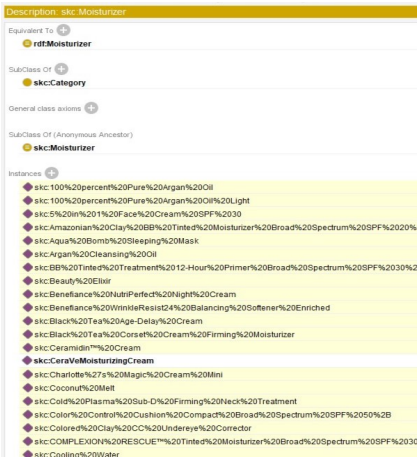
The CSV file found on GitHub [3] was uploaded to the OntoRefine, GraphDB's external tool then the data was edited based on its significance and the RDF mappings were edited and new triples were added. Then in SPARQL the graphs were constructed one by one with CONSTRUCT command and separately downloaded as nine Turtle files. Consecutively, they were all imported to the default graph together with the base ontology turtle file. The same process was performed with the second CSV file that included the Sephora stores' locations. Later on, it was also imported to the default graph. Eventually, the graph with all the datasets could be exported as a turtle file and reused for the further analysis, queries in visualisations and Jupyter notebook.

9 Meaningful Inferences

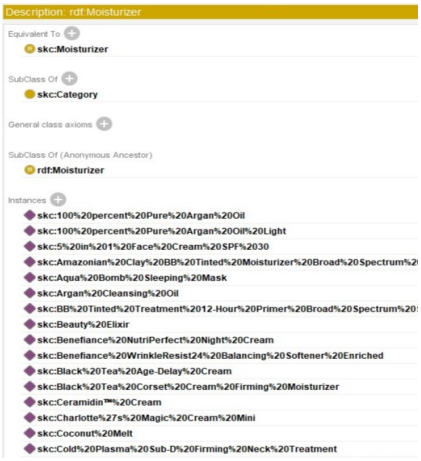
According to the class restrictions in Section 7.6, inferences about the relevant instances were inferred. These include for example all instances of rdf:Moisturizer to be inferred as instances of also skc:Moisturizer as can be seen in Figure 5. The same applies to rdf:Cleanser and skc:Cleanser. It is interesting to see that the before and after picture in Figure 5 have an inverse set of inferred triples, since the two classes are equivalent. Additionally, for example class like Oily_Skin_Products has obtained inferences of instances which have an hasOilyScore of 5.0, as can be seen in Figure 6a and Figure 6b. The same applies for the classes Dry_Skin_Products, Normal_Skin_Products, Combination_Skin_Products.



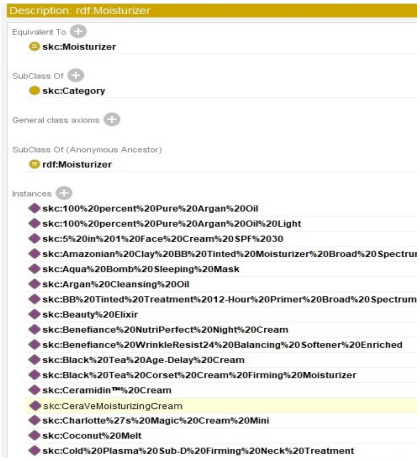
(a) Class "skc:Moisturizer" with no inferencing.



(b) Class "skc:Moisturizer" with inferencing.



(c) Class "rdf:Moisturizer" with no inferencing.

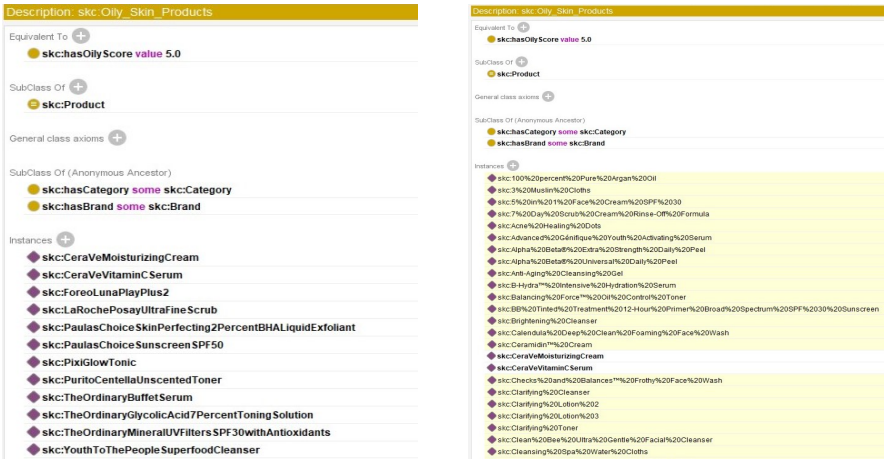


(d) Class "rdf:Moisturizer" with inferencing.

Fig. 5: Inferences in class rdf:Moisturizer and skc:Moisturizer

10 SPARQL Queries

To fulfill the goal of our analysis, as outlined in Section 2, we had to combine the ontology with the two CSV files and develop complex queries to extract knowledge about the products from the raw data. Having data of thousands of individual reviews allowed for analyzing which products are good for which skin types and skin tones. A query was developed to select reviews of only users who have a specific combination of skin type and skin tone (e.g. Oily, Ebony). According to the ratings these users gave to a product the products were



(a) Oily Skin Products with no inferencing (b) Oily Skin Products with inferencing

Fig. 6: Inferences in class Oily Skin Products.

ordered in a list and the top three products became recommended. Additionally, depending on the desired number of steps of the skincare routine (from 1 to 3) the program would select recommended top 3 products of 1-3 categories. For example, if a user wanted a 3 step routine the program would recommend 9 products (3 cleansers, 3 moisturizers and 3 treatments). However, for specific skin type and skin tone combinations there was not enough data to recommend the appropriate number of recommendations. Here came the role of the classes Oily_Skin_Products, Dry_Skin_Products, and so on. We populated these classes by developing queries which extracted the ratings from reviews of users with only a specific skin type and we calculated the average rating for this subclass of users, as can be seen in Figure 12 and example of the OilyScore is presented. The rest can be seen in Appendix 11.

```
skin = Graph()
SKC = Namespace("http://www.semanticweb.org/saraabesova/ontologies/skincare#")
skin.bind("skc", SKC)

#hasOilyScore
results =g.query("""
    PREFIX skc: <http://www.semanticweb.org/saraabesova/ontologies/skincare#>
    PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

    SELECT ?product (AVG(xsd:integer(?rating)) AS ?average)
    WHERE {
        ?reviewID skc:aboutProduct ?product.
        ?reviewID skc:hasSkinType skc:Oily.
        ?reviewID skc:hasRating ?rating.
    } GROUP BY ?product
    """)

for i in results:
    skin.add((i[0], skc.hasOilyScore, i[1]))
```

Fig. 12: Code snippet selecting all instances that should become part of class OilySkinProducts

This information we have added to the graph using data properties (e.g. hasOilyScore). Using class restrictions the graph has deduced that any product that hasOilyScore 5.0 is a Oily_Skin_Product, as seen in Figure 6. This way it was possible to populate these classes, from which we then used some instances to fill missing recommendations for people with unique skin type and skin tone combinations. The complete code with the respective queries and input values can be viewed below in Figure 13 and 14.

```
#INPUT INTERFACE

#Pick your skintype from [Dry, Oily, Combination, Normal]
a="Oily"
#Pick your skintone from [Light, Fair, Medium, Olive, Tan, Porcelain, Deep, Dark, Ebony]
b="Ebony"
#Pick from [1,2,3] depending on how many steps do you want your recommended routine to be
step=3
#Pick from [Australia, Bahrain, Belgium, Brazil, Bulgaria, Canada, China, Czech_Republic, Denmark, Estonia,
sephora_variable="Australia"
```

Fig. 13: Input values for a specific user

```
#Sephora Online and Physical Availability
isephora = "http://dbpedia.org/resource/"+sephora_variable

webpage_results =g.query("""
    PREFIX skc: <http://www.semanticweb.org/saraabesova/ontologies/skincare#>
    PREFIX dbr: <http://dbpedia.org/resource/>

    SELECT ?webpage
    WHERE{
        <"""+isephora+"""> skc:hasSephoraWebPage ?webpage.
    } """)
# Sephora is available online
for result in webpage_results:
    webpage_link = str(result[0])
    if webpage_link:
        print(f"The link to Sephora in your country is here: {webpage_link}")

physical_stores =g.query("""
    PREFIX skc: <http://www.semanticweb.org/saraabesova/ontologies/skincare#>
    PREFIX dbr: <http://dbpedia.org/resource/>

    SELECT ?country ?webpage
    WHERE {
        {?country skc:hasSephoraWebPage ?webpage.
            FILTER CONTAINS (str(?country), "France")}
        UNION {?country skc:hasSephoraWebPage ?webpage.
            MINUS {?country skc:hasSephoraWebPage "https://www.sephora.fr/".}
            MINUS {?country skc:hasSephoraWebPage "https://www.sephora.fr/?country_switch=ca&lang=en".}}
}  """)

# Country link DBpedia
physical_stores_list = []
country = "<" + isephora + ">"

# The physical Sephora store is in the country
for result in physical_stores:
    physical_store = str(result[0])
    physical_stores_list.append(physical_store)
    if country == physical_store:
        print("You can also visit the physical Sephora store in your country")

# If the physical Sephora store is not available in the country
if country not in physical_stores_list:
    # and when there is no webpage results
    if len(webpage_results) == 0:
        print("Unfortunately, Sephora is not available in your country.\n"
              "Try other shops or websites with international shipping:\n"
              "http://www.amazon.com, http://skinstore.com, https://www.cultbeauty.com/ or http://ebay.com")
    else:
        print("Unfortunately, there is no physical Sephora store in your country")
```

Fig. 14: Code that assigns the Sephora webpage links and physical stores

Finally, in addition to outputting the recommended products the local Sephora webpage according to country of residence was to be displayed. Hence, running a query against the triples imported from the SephoraWebPages CSV file the program was able to extract whether the country has a Sephora physical store in the country and output the respective webpage link. The code with the respective queries can be found in Figure 15.

```
#input restructuring
itype="http://www.semanticweb.org/saraabesova/ontologies/skincare#"+a
itone="http://www.semanticweb.org/saraabesova/ontologies/skincare#"+b

steps_list=['Cleanser','Moisturizer','Treatment']
i=0
while i < step:
    print("Top 3 "+steps_list[i]+"s we found for you!")
    products=[]
    result_list=[]

    product_results =g.query("""
        PREFIX skc: <http://www.semanticweb.org/saraabesova/ontologies/skincare#>
        PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

        SELECT DISTINCT ?product
        WHERE {
            ?reviewID skc:aboutProduct ?product.
            ?product rdf:type rdf:"""+steps_list[i]+""".
        }
        """)
    for result in product_results:
        products.append(result[0])

    for product in products:
        chars = ["@", "/", "'", "_", "*", '„', '¯', '"', '©']
        if any([char in strip(product) for char in chars]):
            continue

        ratings=[]
        rating_results =g.query("""
            PREFIX skc: <http://www.semanticweb.org/saraabesova/ontologies/skincare#>
            PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

            SELECT ?rating
            WHERE {
                ?reviewID skc:hasSkinType <"""+itype+""">.
                ?reviewID skc:hasSkinTone <"""+itone+""">.
                ?reviewID skc:hasRating ?rating.
                ?reviewID skc:aboutProduct skc:"""+strip(product)+""".
            }
            """)
        for result in rating_results:
            ratings.append(int(result[0])) ###change to int
        if len(rating_results) !=0:
            average= sum(ratings)/len(ratings)
        else:
            continue

        result_list.append([strip(product),average])

    sorted_list=sorted(result_list, key=itemgetter(1), reverse=True)

    if len(sorted_list) < 3:
        t =g.query("""
            PREFIX skc: <http://www.semanticweb.org/saraabesova/ontologies/skincare#>
            PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

            SELECT ?product
            WHERE {
                ?product rdf:type skc:"""+a+"""_Skin_Products.
                ?product rdf:type rdf:"""+steps_list[i]+""".
            }
            """)
        for j in t:
            sorted_list.append(j)

    recommended_list=[]
    recommended_list.append(sorted_list[0])
    recommended_list.append(sorted_list[1])
    recommended_list.append(sorted_list[2])

    print(recommended_list[0][0])
    print(recommended_list[1][0])
    print(recommended_list[2][0])
    print("")
    i=i+1
```

Fig. 15: Main code deciding on recommending products based on skin type and skin tone

11 Outcomes of SPARQL Queries

From the presented code and queries in Section 10 the program was able to create output for a specific user using their input values. The outcome was 9 recommended products, out of which some used inferences from the SPARQL queries or class restrictions, and the Sephora webpage with potential physical location. An example output in a simple user-interface can be seen in Figure 16. In the above-mentioned example, all 3 recommended cleansers had reviews of the appropriate skin type and skin tone but two moisturizers were imported from class Oily_Skin_Products. The team was under time-pressure hence the output shows a whole URI instead of only a product. Underneath, we can see the output of the link of Sephora webpage and whether is has a physical location.

```
Top 3 Cleansers we found for you!
Clear%20Complexion%20Cleanser
Essential-C%20Toner
Pore-Balance™%20Facial%20Sauna%20Scrub

Top 3 Moisturizers we found for you!
Water%20Drench%20Hyaluronic%20Cloud%20Cream
http://www.semanticweb.org/saraabesova/ontologies/skincare#CeraVeMoisturizingCream
http://www.semanticweb.org/saraabesova/ontologies/skincare#PaulasChoiceSunscreenSPF50

Top 3 Treatments we found for you!
Acne%20Body%20Wash
Breakout%20Fighters
Clearly%20Corrective™%20Dark%20Spot%20Solution

The link to Sephora in your country is here: https://www.sephora.com.au/
Unfortunately, there is no physical Sephora store in your country
```

Fig. 16: Output of the main analysis

12 Limitations and Further Improvements

Despite the team being satisfied with the outcome for the most part, considering the limited duration of the project and data availability, there are certain improvements that could be made to the current analysis and program that could make it stronger. The main 3 improvements are as follows;

1. Random decision factor when selecting a product from one of the product sub-classes (e.g. Oily_Skin_Products). Since all products within the class have a rating 5.0 from the specific skin type group, there is no quantitative way to differentiate between them. Hence, the top 3 products of each product category will always be picked if data is missing rather than choosing different ones every time. A random decision-making factor could make the selection more diverse.

2. Following up on the problem of quantitative comparison making, there can be an occurrence such that e.g. there will be only one review of cleanser

products of a OilySkin and Ebony combination and the rating will be 1.0. In such a case the system would recommend this product and two others from Oily_Skin_Products. It is known that the Oily_Skin_Products have rating 5.0 (that is what classifies them as such). Hence, the first recommended product would be probably less suitable for a specific person than the other two. That is why a quantitative comparison between products found from skin type and tone combination and products from Oily_Skin_Products class. Such a comparison is not easy to make as in the first selection we take two variables into account (e.g. Oily and Ebony), whereas, in the second one we only take one variable (e.g. Oily).

3. Lastly, class restrictions based on ingredients could be developed. This would ensure a more symbolic and chemical approach, compared to the statistical one that is currently employed.

13 Honorary Title: Linked Data Producer

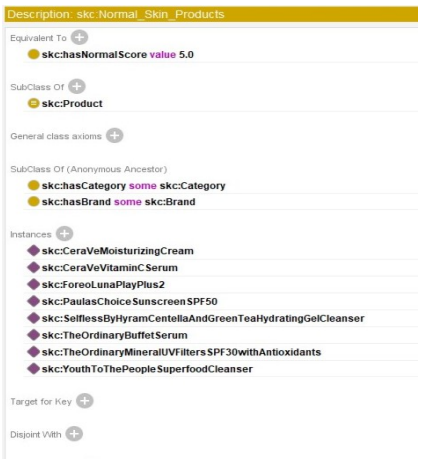
Converting the CSV file which scraped the Sephora website reviews into RDF was a great challenge. Using Ontorefine to make the RDF mappings allowed us to get the base of the relations between classes, these relations can be seen in Appendix 17.

But to merge it with the already existing class required various class restrictions to combine the categories of products which are equivalent or sub-classes of one another. In addition to this, the challenge was to extract more information from the CSV database that could then be transferred into the ontology to possess more non-trivial triples which are more informational and easy to use in the future. These triples in RDF format were developed in Figure 11 and the result of the code can be seen in Figure 18. This result was then imported back to the ontology which created a rich RDF file full of information extracted from the external database.

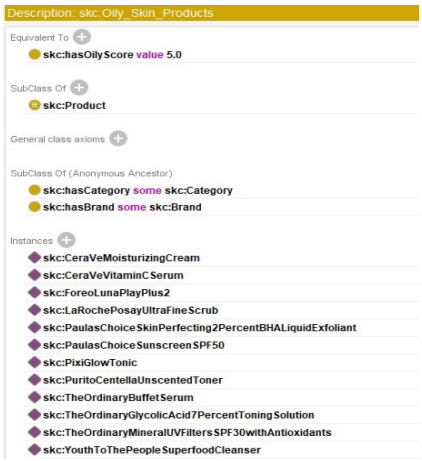
References

1. International (2022), `https://www.sephora.fr/pays.html`
2. Elder, A., Ring, C., Heitmiller, K., Gabriel, Z., Saedi, N.: The role of artificial intelligence in cosmetic dermatology—current, upcoming, and future trends. Journal of Cosmetic Dermatology **20**(1), 48–52 (2021)
3. Gorina, A.: Skincare recommendations (Mar 2022), `https://github.com/agorina91/final_project`
4. Molvar, K.: How to build a skin care routine. The New York Times (2019), `https://www.nytimes.com/guides/tmagazine/skincare-routine`

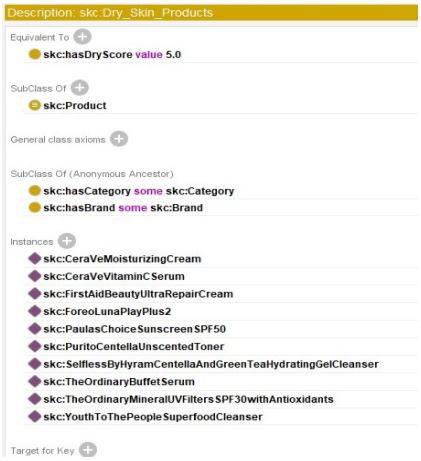
14 Appendices



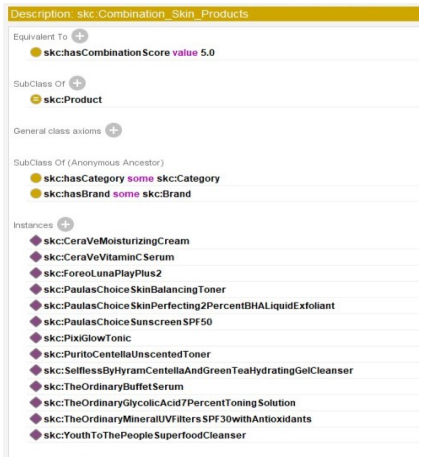
(a) Normal Skin Products



(b) Oily Skin Products

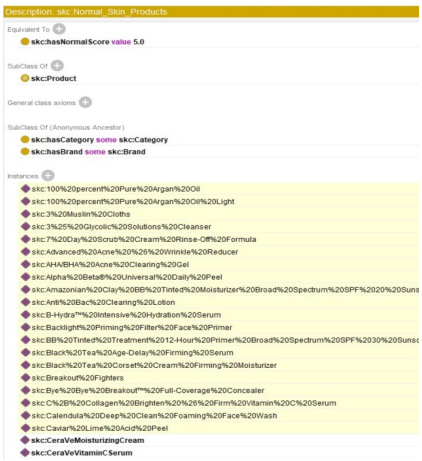


(c) Dry Skin Products

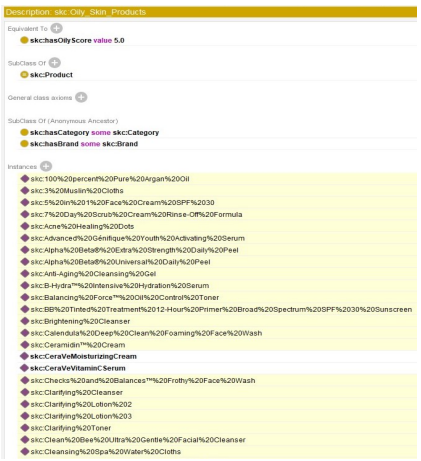


(d) Combination Skin Products

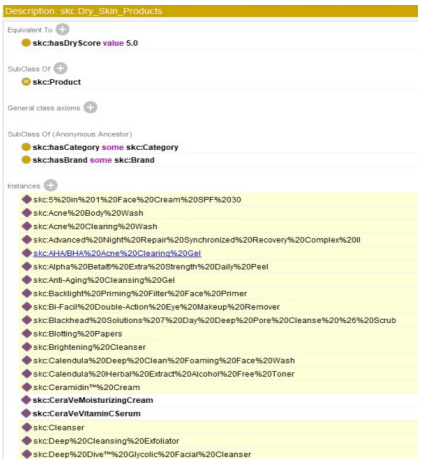
Fig. 7: Class "Skin Type" with no inferencing.



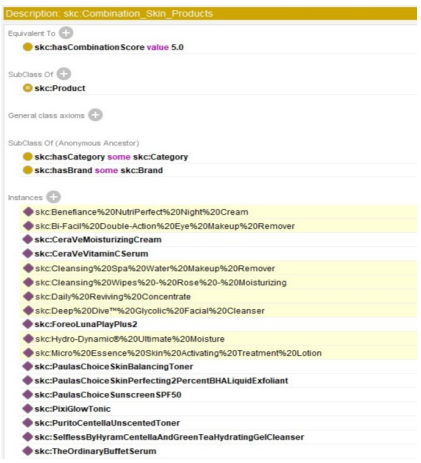
(a) Normal Skin Products



(b) Oily Skin Products

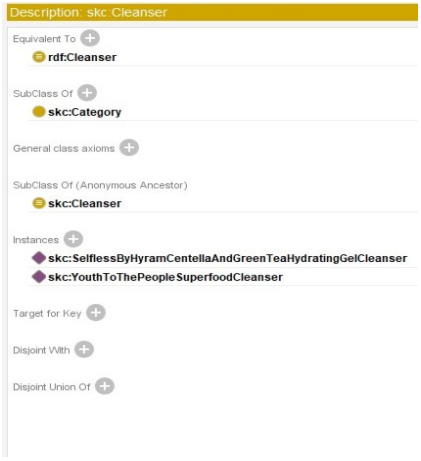


(c) Dry Skin Products

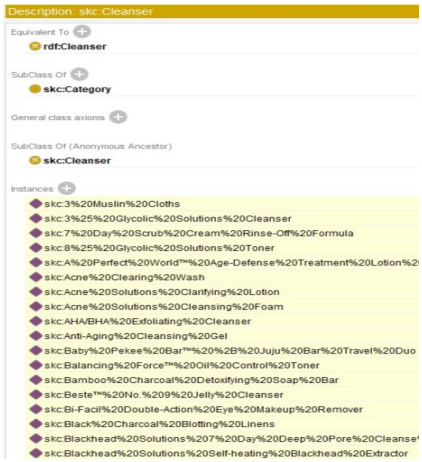


(d) Combination Skin Products

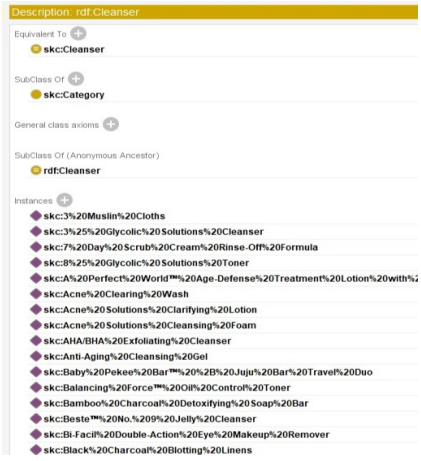
Fig. 8: Class "Skin Type" with inferencing.



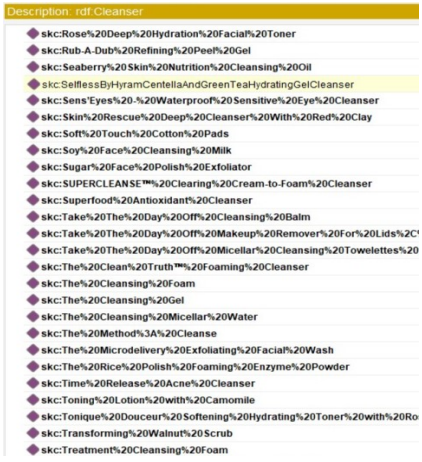
(a) Class "skc:cleanser" with no inferencing.



(b) Class "skc:cleanser" with inferencing.

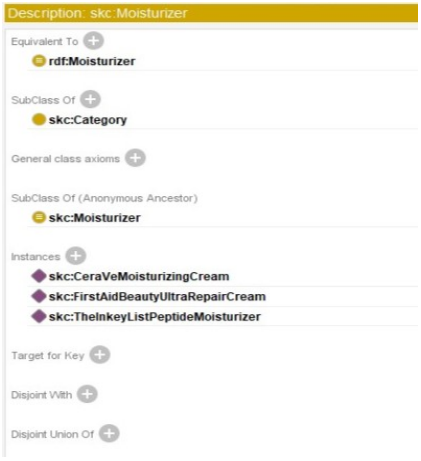


(c) Class "rdf:cleanser" with no inferencing.

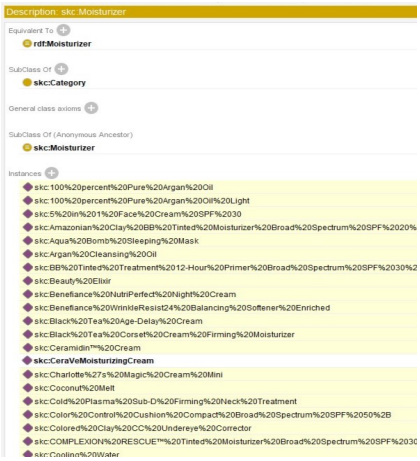


(d) Class "rdf:cleanser" with inferencing.

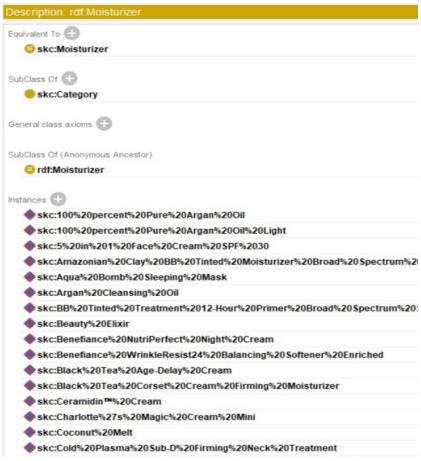
Fig. 9: Inferences in class rdf:Cleanser and skc:Cleanser



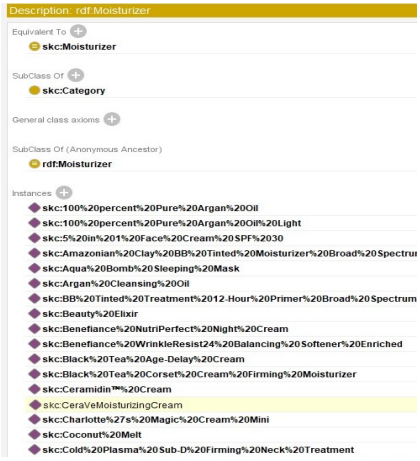
(a) Class "skc:Moisturizer" with no inferencing.



(b) Class "skc:Moisturizer" with inferencing.



(c) Class "rdf:Moisturizer" with no inferencing.



(d) Class "rdf:Moisturizer" with inferencing.

Fig. 10: Inferences in class rdf:Moisturizer and skc:Moisturizer

```
skin = Graph()
SKC = Namespace("http://www.semanticweb.org/saraabesova/ontologies/skincare#")
skin.bind("skc", SKC)

#hasOilyScore
results =g.query("""
    PREFIX skc: <http://www.semanticweb.org/saraabesova/ontologies/skincare#>
    PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

    SELECT ?product (AVG(xsd:integer(?rating)) AS ?average)
    WHERE {
        ?reviewID skc:aboutProduct ?product.
        ?reviewID skc:hasSkinType skc:Oily.
        ?reviewID skc:hasRating ?rating.
    } GROUP BY ?product
    """)

for i in results:
    skin.add((i[0], skc.hasOilyScore, i[1]))

#hasDryScore
results =g.query("""
    PREFIX skc: <http://www.semanticweb.org/saraabesova/ontologies/skincare#>
    PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

    SELECT ?product (AVG(xsd:integer(?rating)) AS ?average)
    WHERE {
        ?reviewID skc:aboutProduct ?product.
        ?reviewID skc:hasSkinType skc:Dry.
        ?reviewID skc:hasRating ?rating.
    } GROUP BY ?product
    """)

for i in results:
    skin.add((i[0], skc.hasDryScore, i[1]))

#hasCombinationScore
results =g.query("""
    PREFIX skc: <http://www.semanticweb.org/saraabesova/ontologies/skincare#>
    PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

    SELECT ?product (AVG(xsd:integer(?rating)) AS ?average)
    WHERE {
        ?reviewID skc:aboutProduct ?product.
        ?reviewID skc:hasSkinType skc:Combination.
        ?reviewID skc:hasRating ?rating.
    } GROUP BY ?product
    """)

for i in results:
    skin.add((i[0], skc.hasCombinationScore, i[1]))
#hasNormalScore
results =g.query("""
    PREFIX skc: <http://www.semanticweb.org/saraabesova/ontologies/skincare#>
    PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

    SELECT ?product (AVG(xsd:integer(?rating)) AS ?average)
    WHERE {
        ?reviewID skc:aboutProduct ?product.
        ?reviewID skc:hasSkinType skc:Normal.
        ?reviewID skc:hasRating ?rating.
    } GROUP BY ?product
    """)

for i in results:
    skin.add((i[0], SKC.hasNormalScore, i[1]))

# serialize_graph(skin)
save_graph(skin, "extra_score_triples.ttl")
```

Fig. 11: Code snippet creating the RDF triples with OilyScore, DryScore, NormalScore, CombinationScore

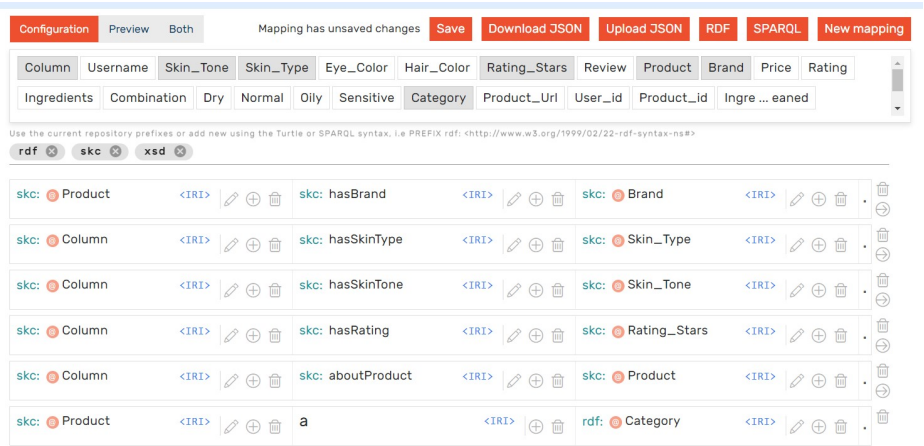


Fig. 17: Ontorefine RDF mappings

```
@prefix skc: <http://www.semanticweb.org/saraabesova/ontologies/skincare#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://www.semanticweb.org/saraabesova/ontologies/skincare#%2BRetinol%20Vita%20C%20Power%20Serum> skc:hasCombinationScore 3.625
;
    skc:hasDryScore 4.16666666666666666666666667 ;
    skc:hasNormalScore 3.5 .

skc:100%20percent%20Pure%20Argan%20Oil skc:hasCombinationScore 3.0909090909090909090909091 ;
    skc:hasDryScore 1.0 ;
    skc:hasNormalScore 5.0 ;
    skc:hasOilyScore 5.0 .

skc:100%20percent%20Pure%20Argan%20Oil%20Light skc:hasCombinationScore 4.5 ;
    skc:hasDryScore 3.8 ;
    skc:hasNormalScore 5.0 ;
    skc:hasOilyScore 2.0 .

skc:3%20Muslin%20Cloths skc:hasCombinationScore 4.384615384615384615384615385 ;
    skc:hasDryScore 3.0 ;
    skc:hasNormalScore 5.0 ;
    skc:hasOilyScore 5.0 .
```

Fig. 18: RDF generated files with extra triples